Vers une analyse syntaxique à granularité variable

Tristan Vanrullen

▶ **To cite this version:**

Tristan Vanrullen. Vers une analyse syntaxique à granularité variable. Travaux Interdisciplinaires sur la Parole et le Langage, 2003, 22, pp.186-212. hal-00134200

HAL Id: hal-00134200
https://hal.science/hal-00134200

Submitted on 1 Mar 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

VERS UNE ANALYSE SYNTAXIQUE A GRANULARITÉ VARIABLE

Tristan Van Rullen

Résumé

Il est souhaitable qu'une analyse syntaxique -en traitement automatique des langues naturelles- soit réalisée avec plus ou moins de précision en fonction du contexte, c'est-à-dire que sa granularité soit réglable. Afin d'atteindre cet objectif, nous présentons ici des études préliminaires permettant d'appréhender les contextes technique et scientifique qui soulèvent ce problème. Nous établissons des directions épistémologiques pour les développements à réaliser et pour leur évaluation. Puis nous décrivons un jeu d'expériences menées dans ce cadre, sur des algorithmes basés sur un formalisme de satisfaction de contraintes (celui des Grammaires de Propriétés) ayant l'avantage de permettre l'utilisation des mêmes ressources linguistiques avec un degré de précision réglable. Deux types de granularité sont définis. Enfin, nous envisageons les développements ultérieurs pour une analyse syntaxique à granularité variable.

Mots-clés : théorie linguistique, syntaxe, analyse automatique, grammaire de propriétés, contraintes, granularité, complexité, algorithmes.

Abstract

It is gainful for a syntactic analysis - in Natural Language Procession- to be carried out with more or less accuracy according to the context, i.e. its granularity should be adjustable. In order to reach this objective, we present here preliminary studies allowing, first of all, to understand the technical and scientific contexts which raise this problem. We establish an epistemological framework for the developments to be carried out and for their evaluation. We then describe an experiment set carried out within this framework, on algorithms based on a constraints satisfaction formalism (Grammars of Properties) which allows the use of the same linguistic resources with an adjustable degree of accuracy. Two kinds of variable granularity are defined. Lastly, we consider further developments towards a syntactic analysis with variable granularity.

Keywords : linguistic theory, parsing, property grammars, constraints, granularity, complexity, algorithms.

VAN RULLEN, Tristan, (2003), Vers une analyse syntaxique à granularité variable, *Travaux Interdisciplinaires du Laboratoire Parole et Langage,* vol. 22, p. 185-212.

Introduction

Certaines applications en Traitement Automatique des Langues Naturelles s'appuient sur des techniques d'analyse superficielle de la syntaxe (typiquement celles traitant des données volumineuses), d'autres comptent sur l'analyse profonde (par exemple la traduction automatique). Les techniques employées dans ces deux cas sont tout à fait différentes. La première se fonde habituellement sur des méthodes stochastiques, la seconde sur des techniques symboliques. Cependant, ceci peut constituer un problème pour des applications qui auraient besoin d'une analyse peu profonde la plupart du temps et dans certaines situations d'une information plus détaillée. C'est typiquement le cas pour les systèmes de synthèse vocale. De telles applications se fondent habituellement sur des analyseurs peu profonds afin de calculer les groupes intonatifs sur la base d'unités syntaxiques (ou plus précisément sur des 'chunks'). Mais dans certains cas, une information aussi superficielle n'est pas assez précise. Une solution consisterait alors à employer une analyse profonde pour quelques constructions. Aucun système mettant en application une telle approche n'existe. Cela est dû en particulier au fait qu'un tel système exigerait deux traitements différents, le second refaisant en fait la totalité du travail. De fait, il est difficile d'imaginer, dans le cadre génératif classique, de mettre en application une technique d'analyse capable de calculer des 'chunks', et dans certains cas, des expressions plus complexes avec une organisation hiérarchique.

Nous présentons ici un formalisme fondé sur les contraintes, qui constitue une réponse possible à ce problème. Cette approche permet l'utilisation d'une même ressource linguistique (c.-à-d. une grammaire unique) qui puisse être employée entièrement ou partiellement par l'analyseur. Cette approche se fonde sur le fait que (1) toute l'information linguistique est représentée au moyen de contraintes et (2) les contraintes sont de type régulier. L'idée consiste alors à mettre en application une technique qui puisse se servir de quelques contraintes dans le cas de l'analyse peu profonde, et de leur ensemble entier pour l'analyse profonde. Dans notre formalisme, les contraintes sont organisées en différents types. Régler la granularité de l'analyse réside alors dans le choix des types de contrainte devant être satisfaits.

La première partie de cet article propose une réflexion sur la position de ce problème au sein du Traitement Automatique des Langues Naturelles, tant des points de vue technique et scientifique qu'épistémologique, notamment sur la question de l'évaluation qui soulève une problématique particulière : comment évaluer un travail en l'absence de référence ?

La seconde partie aborde le formalisme des Grammaires de Propriétés, ses avantages principaux en termes de représentation et d'exécution ; nous y exposons le choix d'utiliser ce formalisme pour les algorithmes de nos expériences.
La troisième partie illustre les caractéristiques respectives des différentes techniques choisies pour nos expériences en décrivant pour le même exemple les conséquences du réglage de la granularité. Nous donnons une évaluation quantitative de la complexité des algorithmes choisis.
Une quatrième partie propose deux expériences portant sur l'application de ces techniques à un vaste corpus journalistique -quoique susceptibles de s'appliquer à d'autres types d'analyses utilisées en 'shallow parsing' et en 'deep parsing'. Nous y effectuons une évaluation relative de ces techniques par le multiplexage logique de leurs sorties, procédé qui, selon nous, répond à notre problématique dans les limites du cadre que nous nous sommes fixés.
Enfin, nous concluons en présentant quelques perspectives pour les travaux à venir.

1. Position du problème

Nous nous proposons ici de situer la question de l'analyse syntaxique au sein du Traitement Automatique des Langues Naturelles et de définir le problème de la granularité qui y apparaît.

1.1. Des analyseurs syntaxiques

Toute analyse syntaxique, si elle peut procéder de diverses manières, effectue **un découpage de texte en blocs**. Ces blocs sont les groupes syntaxiques valides pour une syntaxe donnée.

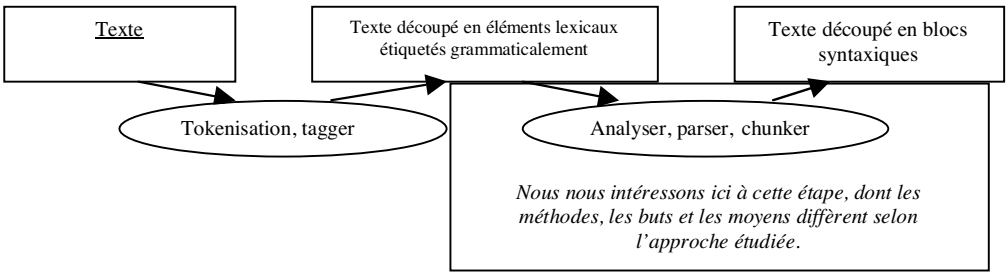


Figure 1
Position de l'analyse syntaxique dans le Traitement Automatique des Langues Naturelles

Le contexte d'une analyse syntaxique se définit avec les paramètres suivants :

- la théorie syntaxique elle-même,
- la tolérance de ces algorithmes aux ambiguïtés (on dira qu'un algorithme s'arrêtant à une seule interprétation est déterministe),
- l'algorithme d'analyse (la technique employée pour découper le texte),

Les variables, en analyse syntaxique, sont :

- la couverture de ces algorithmes (la quantité de textes pouvant être traités, la qualité exigée de ces textes en termes de grammaticalité, de normativité, pour qu'ils puissent être traités),
- l'efficacité de ces algorithmes (l'exhaustivité des résultats obtenus),
- leur qualité (justesse de l'analyse),
- leur complexité (rendant compte de la vitesse d'obtention des résultats en fonction de la taille des données à traiter.)

Le schéma de la page suivante résume les approches les plus courantes.
Citons, à titre d'illustration, quelques analyseurs correspondant à chacune de ces approches :

- Approche stricte :

  Cascade d'automates à états finis, J.-P. Chanod, Rank Xerox Center. (voir [Cha00], [ChaMok96], [ChaTap96] et [ChaTap97]).

- Approche statistique :

  Le programme stochastique de Church, (cf. [Chu88]), basé sur les n-grammes et les chaines de Markov.

- Approche superficielle :

  shallow parsers, chunkers : les programmes de Liberman et Church ([LibChu92]), d'Abney ([Abn91]), de Bourigault (1994), de Di Cristo ([DiC98] et [Dic&Al01]).

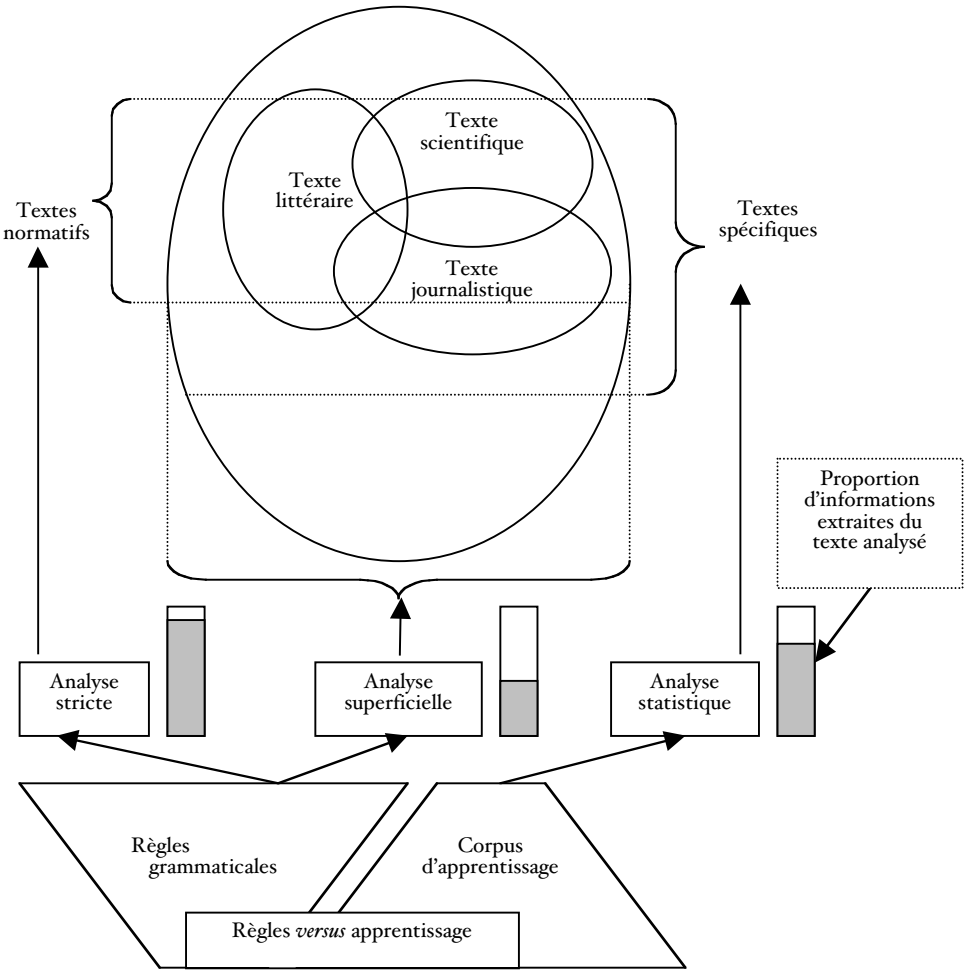


Figure 2
Les types de traitement existants et leur portée

1.2. Différentes exigences selon l'application

La pléiade des programmes qui se proposent d'analyser des textes répond à des exigences différentes, conditionnées par les besoins (pratiques, rapides, à large couverture, ou bien précis, complets, détaillés) des applications utilisant leurs résultats.
Nous distinguerons à ce propos quelques domaines d'utilisation :

- Recherche automatique d'informations en langues naturelles.
- Mise en forme automatique de texte.
- Transformation de texte (synthèse, résumé, traduction.)
- Synthèse vocale.
- Un cumul de tous ces traitements est parfois souhaité.

1.3. Des directions épistémologiques pour nos démarches

La rigueur qu'exige le travail de recherche ne doit pas être contrariée par les contraintes qu'imposent les applications. De fait, les variables que nous avons énumérées ci-dessus montrent à quel point chaque choix théorique et algorithmique aura des conséquences sur les résultats. Un programme deviendrait vite un gadget si nous oubliions le contexte qui l'a vu naître.
Afin de contrôler ces variables à chaque étape de nos développements, il est nécessaire de se donner un cadre épistémologique pour nos démarches. Ce cadre, déplaçable pour tout projet scientifique, trouve un sens dans la définition de l'objet étudié et permet d'orienter les développements.

L'objet de notre étude est l'analyse syntaxique automatique, au sens le plus ouvert.

De manière formelle, prenons un peu de recul par rapport à l'objet de notre étude.
1 : Une théorie scientifique, susceptible d'évoluer, se place derrière chaque formalisme que nous pourrions adopter.
2 : Une axiomatique qui ne doit pas rester implicite guide les évolutions de plusieurs théories ayant le même objet. Cette axiomatique est la supposition de cet objet. Elle admet une logique qui est indépendante des évolutions des théories sous-jacentes. Cette logique et ses théorèmes sont le seul contexte dans lequel nous pouvons affirmer qu'un programme est valide.
3 : Notre démarche doit rester autant que possible indépendante de la théorie, mais non de l'axiomatique. Si plusieurs tendances théoriques s'opposent et se contredisent, un choix s'avère alors nécessaire parmi celles-ci pour permettre des développements pratiques.

D'autre part, du point de vue informatique, le terme *syntaxe* ne désigne pas le même objet qu'en linguistique. En Traitement Automatique des Langues Naturelles, ce double héritage théorique (informatique et linguistique) nous amène à considérer le mot *syntaxe*

avec précision. Nous posons immédiatement le présupposé suivant qui en donnera le contour pour nos travaux :

La syntaxe est caractérisée par les éléments que sont :

-les *syntagmes*, repérés comme parties du *texte*. Un syntagme est lui-même constitué de *mots* (composés ou non), auxquels sont associées des *catégories*.

-Les *catégories syntaxiques*, associées à chaque syntagme ou à chaque mot.

Nous pourrions adopter une autre axiomatique de la syntaxe (par exemple en définissant le syntagme à partir de mots simples et en ne reconnaissant pas les mots composés), ce qui rendrait immédiatement caduques tous les choix effectués ci-dessous et dans la suite de nos travaux. Une révision de cette base nécessiterait la refonte totale des théories et outils développés ici.

Dans cette acception, l'*analyse syntaxique* qui pourra être effectuée couvrira non seulement la *syntaxe* au sens linguistique, mais aussi toute forme d'analyse d'informations pouvant être transcrites (analyse prosodique, sémantique, pragmatique etc...). Cet élargissement du sens découle de l'approche informatique inaugurée par la théorie des langages et la compilation des programmes. Les objets manipulés en deçà de ce terme, syntagmes et catégories – quoique ces termes soient empruntés au vocabulaire de la linguistique-, sont à définir pour chaque type d'analyse voulue (syntaxique, prosodique, sémantique etc...)

Nous proposons donc les éléments suivants :

Toute théorie du texte qui en suggérerait un découpage en *groupes d'informations* dotés d'une *catégorie* sera appelée dans cet article et dans nos travaux une *syntaxe*.

Ainsi, la syntaxe et les autres domaines de la linguistique n'ont pas forcément à être en concurrence au moment du traitement du texte, mais peuvent simultanément être considérés comme des théories fondées sur des axiomatiques du découpage textuel en unités pertinentes, que par abus de langage nous appellerons syntaxe.

La superposition des unités obtenues par deux théories fondées sur une même axiomatique syntaxique n'est pas garantie, c'est le cadre dans lequel nous devons travailler. (cf. à ce propos [DiC98] et [Dic&Al01])

Nous pouvons à présent définir le problème dont traite cet article.

1.4. Le problème de la granularité

Tentons de reformuler les contraintes qui orientent notre travail. Certaines applications ont besoin la plupart du temps d'une analyse peu profonde et dans certaines situations d'une information plus détaillée. C'est typiquement le cas pour les systèmes de synthèse vocale. De telles applications se fondent habituellement sur des analyseurs peu profonds afin de calculer les groupes intonatifs sur la base d'unités syntaxiques (ou plus précisément sur des 'chunks'), en accord avec les recommandations d'auteurs de tels systèmes comme Allen *et al.* [All&Al87] :

> *« Un système de synthèse vocale demande des choses plutôt inhabituelles à un analyseur. Il doit avoir une large (quoique légère autant que possible) couverture des textes, sans restriction, plutôt qu'une analyse profonde d'un domaine restreint. L'échec de l'analyse est inacceptable dans un système TTS. »*

Dans certains cas, une information trop superficielle n'est pas assez précise. Une solution consisterait alors à employer une analyse profonde pour quelques constructions. Aucun système mettant en application une telle approche n'existe. C'est dû en particulier au fait que ceci exigerait deux traitements différents, le second refaisant en fait la totalité du travail. De fait, il est difficile d'imaginer dans le cadre génératif classique de mettre en application une technique d'analyse capable de calculer des 'chunks', et dans certains cas, des expressions plus complexes avec une organisation hiérarchique.

Notre problème est celui-ci : comment développer un outil d'analyse tantôt superficiel, tantôt profond, basé sur les mêmes ressources linguistiques, pouvant s'adapter à de nouvelles ressources et pouvant intégrer plusieurs interprétations pour une même entrée ?

Découpons ce problème en deux parties :

-Il nous faut un *outil d'analyse à granularité variable*.
Cet outil permettra de réaliser une analyse plus ou moins profonde en fonction du contexte.

-Il nous faut un outil capable d'effectuer une *sélection de granularité entre différentes techniques.*
Cet outil permettra de gérer des interprétations concurrentes pour un même texte (par exemple un contour syntaxique et un contour prosodique, au sens linguistique).

1.5. Problématiques émergentes

Dans les développements à venir, nous ne devons pas oublier qu'il s'agit de traiter de front un ou plusieurs formalismes linguistiques, des moyens et des théories appartenant au

domaine de l'informatique, ainsi que des attentes appartenant aux deux domaines. Cette exigence doit être un guide pour tous nos développements.
Une question vient immédiatement : comment évaluer les outils qui seront développés ? Si nous disposions d'un corpus de référence, étiqueté selon des critères précis, l'évaluation de notre système serait un détail. Ici, en l'absence de corpus de référence et sachant qu'il n'en existera pas un qui puisse être seul référent pour toutes les théories que nous voulons pouvoir traiter, il nous faut une fois de plus se poser une question épistémologique : qu'est-ce qu'évaluer notre système ?
Sans répondre immédiatement, nous pouvons garder cette problématique en mémoire afin qu'elle éclaire les développements qui suivent.

2. Les Grammaires de Propriétés comme formalisme candidat pour nos exigences

La notion de contraintes a une importance cruciale en linguistique, voir par exemple [Mar90], [PolSag94], [SagWas99]. Les théories récentes (du paradigme de l'analyse par contraintes à celui des Principes et Paramètres) se fondent sur cette notion. Un des intérêts principaux de l'emploi des contraintes réside dans la possibilité de représenter n'importe quel genre d'information (très générale aussi bien que locale ou contextuelle) à l'aide d'un dispositif unique. Nous présentons dans cette section un formalisme, appelé Grammaires de Propriétés, décrit en [BesBla99] ou [Bla01], qui permet de concevoir et de représenter toute l'information linguistique en termes de contraintes sur des objets linguistiques. Dans cette approche, les contraintes sont vues comme des relations entre deux (ou plus) objets : il est possible de représenter l'information de manière horizontale. La première étape dans ce travail revient à identifier les relations habituellement utilisées en syntaxe.
Ceci peut être fait empiriquement et nous suggérons, en adaptant une proposition de Bès (1999), l'ensemble de contraintes suivantes : linéarité, dépendance, obligation, exclusion, exigence et unicité. Dans une perspective de structure syntagmatique, toutes ces contraintes participent à la description d'un syntagme. La figure suivante esquisse leurs rôles respectifs, illustrés avec quelques exemples pour le SN.

Contrainte	Définition
Linéarité (<)	Précédence linéaire
Dépendance (→)	Relation de Dépendance entre catégories
Obligation (*Oblig*)	Ensemble de catégories obligatoires et uniques. Une de ces catégories (et seulement une) doit être réalisée dans un syntagme.
Exclusion (≠)	Restriction de cooccurrence entre des ensembles de catégories
Exigence (⇒)	Cooccurrence obligatoire entre des ensembles de catégories
Unicité (*Uniq*)	Ensemble des catégories qui ne peuvent être répétées dans un syntagme

Par cette approche, décrire un syntagme revient à indiquer l'ensemble des contraintes couvrant les catégories qui peuvent le constituer. Une contrainte se présente comme suit. Soit R un symbole représentant une relation de contrainte entre deux (ensembles de) catégories. Une contrainte de la forme a R b stipule que si a et b sont réalisés, alors la contrainte a R b doit être satisfaite. L'ensemble des contraintes décrivant un syntagme peut être représenté comme un graphe reliant plusieurs catégories. L'exemple suivant illustre quelques contraintes pour le SN.

Linéarité	*Det < N; Det < AP;* *AP < N; N < PP*
Exigence	*N[com] ⇒ Det*
Exclusion	*N ≠ Pro; N[prop] ≠ Det*
Dépendance	*Det → N; AP → N; PP → N*
Obligation	*Oblig(NP) = {N, Pro, AP}*

Dans cette description, on peut noter par exemple une relation d'*exigence* entre le nom commun et le déterminant (une telle contrainte met en application la relation de complémentation) ou une contrainte d'*exclusion* qui indiquent la restriction de cooccurrence entre un nom et un pronom ou un nom propre et un déterminant. On peut noter l'utilisation de traits : c'est habituellement le cas dans les théories linguistiques ; une catégorie a plusieurs propriétés qui peuvent être héritées quand la description de la catégorie est raffinée

(dans notre exemple, le type *nom* a deux sous-types, *propre* et *commun*). Toutes les contraintes impliquant un nom impliquent également ses sous-types. En conclusion, la relation de dépendance, qui est sémantique, indique que le dépendant (actant) doit accorder ses traits sémantiques avec le gouverneur. Comme HPSG le fait maintenant avec le dispositif de DEPS tel que décrit dans [Bou&Al01], cette relation concerne toute catégorie, et pas nécessairement celle qui est gouvernée. De cette façon, la différence entre un complément et une adjonction est que seul le complément est choisi par une contrainte d'exigence, les deux étant contraints par une relation de dépendance. Ceci signifie également qu'une différence peut être faite entre la tête syntaxique (indiquée par la contrainte d'obligation) et la sémantique (le gouverneur de la relation de dépendance), même si dans la plupart des cas, ces catégories sont les mêmes. D'autre part, on peut imaginer de spécifier des dépendances dans un syntagme entre deux catégories autres que la tête (voir à ce propos [DucDeb01]).

Un des avantages principaux de cette approche est que les contraintes forment un système et que toutes les contraintes sont au même niveau. À la différence d'autres approches comme la théorie d'optimalité, présentée dans [PriSmo93] et [ArcLan87], il n'existe aucune hiérarchie entre elles et on peut choisir, selon les besoins, de vérifier l'ensemble entier de contraintes ou seulement un sous-ensemble. Dans cette perspective, employer une technique de satisfaction de contraintes comme base pour la stratégie d'analyse permet de mettre en application la possibilité de vérifier seulement un sous-ensemble de ce système de contraintes. Ce qui est intéressant est que certaines contraintes comme la linéarité fournissent des indications en termes de frontières, comme décrit par exemple dans [BlaMor90]. Il s'ensuit que la vérification de ce sous-ensemble de contraintes peut constituer une technique de parenthésage. La vérification de plus de contraintes en addition à la linéarité permet de raffiner l'analyse. Enfin, la même technique d'analyse (satisfaction de contraintes) peut être employée pour des analyses peu profondes et des analyses profondes. Pour être plus précis, en utilisant les mêmes ressources linguistiques (lexique et grammaire), nous proposons une technique qui permette de choisir la granularité de l'analyse.

3. Granularité variable. Algorithmes opérant à des profondeurs différentes, fondés sur le même formalisme

Nous pouvons à présent observer deux techniques basées sur le formalisme décrit ci-dessus. Une troisième technique, adaptée de l'algorithme Chink/Chunk de Philippe Di Cristo est présentée à fin de référence et de comparaison pour la quantification de la complexité des algorithmes aussi bien que pour montrer les différences entre les résultats obtenus.

Ces techniques n'ont pas ici la prétention de fondre en un seul outil les différents niveaux d'analyse, mais de démontrer pour l'instant les possibilités du formalisme choisi, qui permet des analyses de profondeur différente avec les mêmes ressources. D'autres détails à propos des algorithmes présentés ici et de leur évaluation pourront être lus dans [BlaVan02] et [Bla&Al02].

3.1. Description des algorithmes

Les algorithmes que nous avons développés sont décrits ici en termes techniques et un exemple basé sur l'analyse de la phrase ci-dessous illustre leur fonctionnement.

> *"Le compositeur et son librettiste ont su créer un équilibre dramatique astucieux en mariant la comédie espiègle voire égrillarde et le drame le plus profond au cœur des mêmes personnages."*

3.1.1. Un algorithme Chink/Chunk pour référence

Dans le but de comparer nos techniques, nous avons établi un chunker robuste et simple. Ce programme rapide donnera une idée de la complexité minimale pour les deux techniques basées sur des Grammaires de Propriétés. Cet algorithme se fonde sur la technique de Liberman et de Church (voir[LibChu92]) et sur le chunker de Di Cristo (voir [DiC98] et [DiC&Al00]). Son mécanisme consiste en une segmentation de l'entrée en chunks, à l'aide d'un automate à états finis -se servant de *mots fonction* comme frontières de bloc. Une amélioration de la notion de chunk est mise en application, en utilisant des conjonctions comme éléments neutres pour chunks en cours de construction. Cet algorithme donne déjà des résultats intéressants pour calculer des groupes prosodiques dans le synthétiseur vocal SYNTAIX.
Ce premier exemple montre une représentation non hiérarchique de la phrase découpée en chunks. Aucune information linguistique n'est fournie.

```
[ (phrase)
  [ (chunk)Le compositeur et son librettiste ont su créer]
  [ (chunk)un équilibre dramatique astucieux]
  [ (chunk)en mariant]
  [ (chunk)la comédie espiègle]
  [ (chunk)voire égrillarde]
  [ (chunk)et le drame]
  [ (chunk)le plus profond]
  [ (chunk)au coeur des mêmes personnages]]
```

Sortie de l'analyseur Chink/Chunk

3.1.2. Un analyseur superficiel et déterministe basé sur les Grammaires de Propriétés

Par cette technique, nous obtenons une information hiérarchique et grammaticale tout en préservant la robustesse et l'efficacité du traitement. Dans cette perspective, nous nous servons d'une grammaire représentée dans le formalisme de grammaire de propriété décrit ci-dessus. Un des intérêts principaux de ce formalisme est qu'il ne se sert pas réellement de la notion de grammaticalité, la remplaçant par un concept plus général de caractérisation. Il devient alors possible de proposer une description en termes de propriétés syntaxiques pour n'importe quel genre d'entrée (grammaticale ou non). L'ouverture et la fermeture des chunks se fonde ici sur une information compilée à partir de la grammaire. Cette information consiste en un ensemble de coins gauches et coins droits potentiels, ainsi que les constituants potentiels pour les chunks. Elle est obtenue en compilant les propriétés de linéarité, d'obligation, d'exigence et d'exclusion décrites dans les sections précédentes ainsi que, indirectement, une propriété de constituence. À partir de cette grammaire compilée, l'analyseur fonctionne de la manière suivante :

Deux piles, une des catégories ouvertes et une seconde des catégories fermées, sont construites après l'analyse de chaque nouveau mot parcouru : nous pouvons ouvrir de nouvelles catégories ou fermer des catégories déjà ouvertes en appliquant quelques règles. Cet algorithme étant récursif, les actions d'ouverture, de continuation et de fermeture sont elles aussi récursives. C'est la raison pour laquelle les règles doivent avoir une définition stricte afin d'être certain que l'algorithme sera déterministe et se terminera toujours. Cette technique superficielle d'analyse peut être vue comme ensemble de règles de production/réduction/coupure.

• Règle 1 : Ouvrez une expression p pour la catégorie courante c si c peut être le coin gauche de p.

• Règle 2 : N'ouvrez pas une catégorie déjà ouverte si la catégorie appartient au syntagme courant ou est son coin droit. Autrement, nous pouvons la rouvrir si le mot courant peut seulement être son coin gauche

• Règle 3 : Fermez les syntagmes ouverts si le syntagme le plus récemment ouvert ne peut ni continuer l'un d'entre eux, ni être l'un de leurs coins droits.

• Règle 4 : Lors de la fermeture d'une expression, appliquez les règles 1, 2 et 3. Ceci peut fermer ou ouvrir de nouvelles expressions prenant en compte toutes les catégories y compris celles des syntagmes.

```
[ (phrase)
  [ (NP)Le compositeur
    [ (AP) et]
    son librettiste]
  [ (VP)ont su créer]
  [ (NP) un équilibre
    [ (AP)dramatique astucieux]]
  [ (Compl)en
    [ (VP)mariant]]
  [ (NP)la comédie
    [ (AP)espiègle voire égrillarde et]
    le drame
    [ (Sup)le plus profond]]
  [ (PP)au cœur de
    [ (NP)les
      [ (AP)mêmes]
      personnages]]]
```

Sortie de l'analyseur superficiel

Ce deuxième exemple donne une représentation hiérarchique de la phrase, divisée en chunks portant une étiquette. Comme nous avons employé une version précompilée de la grammaire (raccourcie) et parce que nous avons forcé quelques choix syntaxiques afin de préserver le déterminisme et la terminaison de l'analyse, il s'avère que des erreurs ont été faites par l'analyseur superficiel : les conjonctions sont distinguées. Malgré ces lacunes, la coupure est améliorée et la plupart des catégories sont détectées correctement

3.1.3. Un analyseur profond et non déterministe basé sur les Grammaires de Propriétés

L'analyse profonde est directement basée sur les Grammaires de Propriétés. Elle consiste, pour une phrase donnée, à établir tous les sous-ensembles possibles d'éléments juxtaposés qui puissent décrire une catégorie syntaxique. Un sous-ensemble est clairement caractérisé s'il satisfait les contraintes de la grammaire. Ces sous-ensembles s'appellent des *arcs*, ils décrivent un segment de la phrase entre deux positions.

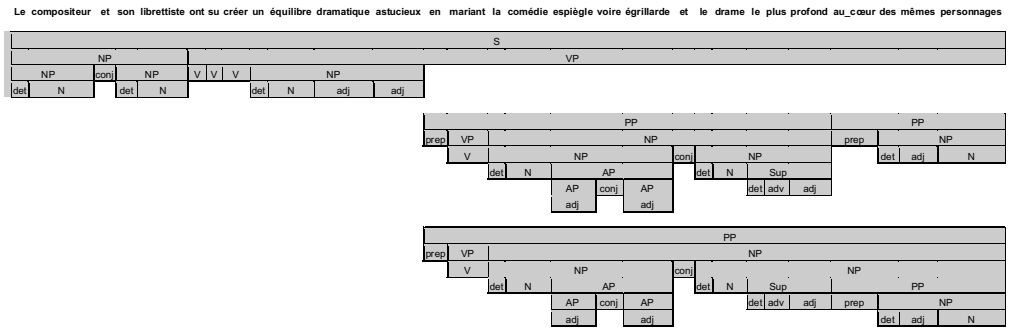
À la première étape, chaque catégorie lexicale est considérée comme arc du niveau 0. La phase suivante revient à produire tous les sous-ensembles possibles d'arcs au niveau 0. Le résultat est un ensemble d'arcs de niveau 1. Les étapes suivantes fonctionnent de la même manière et produisent tous les sous-ensembles possibles d'arcs, chaque étape correspondant à un niveau. L'algorithme finit quand aucun nouvel arc ne peut être créé.

Un arc est caractérisé par :

- Une position initiale et finale dans la phrase,
- Une catégorie syntaxique,

- Un ensemble de propriétés syntaxiques décrivant la catégorie
- Un ensemble de constituants : un unique constituant lexical au niveau 0, et un ou plusieurs arcs aux autres niveaux.

Le dernier exemple (figure ci-dessous) présente deux des phrases de couverture maximale produites par l'analyseur profond. Cette figure, qui illustre l'ambiguïté d'attachement de pp, montre une structure hiérarchique. Reste que chaque étiquette représente l'état du système de contraintes après évaluation.



Sortie de l'analyseur profond

3.2. Résultats et analyse de complexité

Outre les résultats de l'analyse, qui nous intéresseront dans la prochaine section, nous avons un objectif de rapidité à contrôler. Un calcul de complexité a été réalisé pour chacune des techniques.

3.2.1. Résultats pour Chink/Chunk

L'algorithme Chink/Chunk est une manière simple mais efficace de détecter les frontières syntaxiques. Dans les cas moyens, optimaux et pires, pour M phrases, chaque phrase se composant de Nm mots, sa complexité a un ordre de M*Nm*Constante. C'est-à-dire une complexité linéaire.

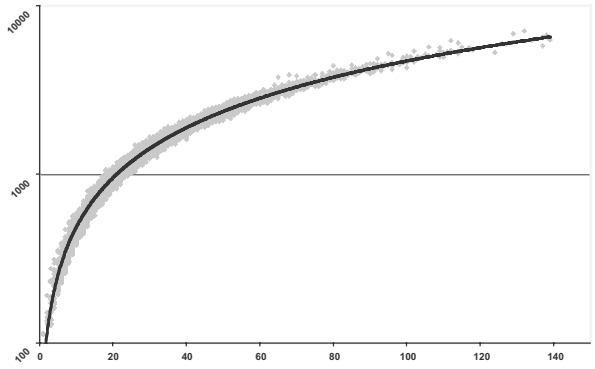


Figure 3
Instructions / nombre de mots
Pour Chink/Chunk (échelle logarithmique)

3.2.2. Résultats pour l'analyseur superficiel

Avec l'algorithme d'analyse superficielle, nous pouvons détecter et annoter plus de données syntaxiques et hiérarchiques : dans les cas moyens, pires et meilleurs, pour M phrases, chaque phrase se composant de Nm mots; pour un ensemble de C catégories précompilées, sa complexité est de l'ordre de M*C*(Nm_+Nm)*Constante. C'est-à-dire une complexité polynômiale.

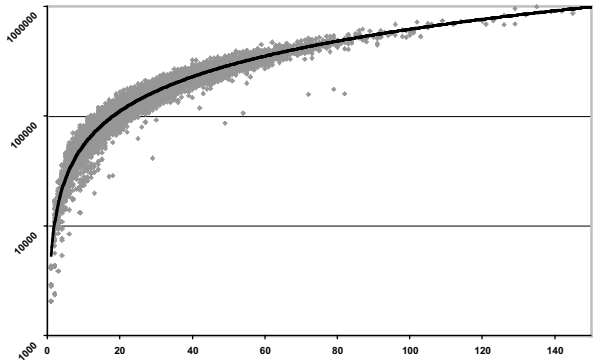


Figure 4
Instructions / nombre de mots
pour l'anayseur superficiel (échelle logarithmique)

3.2.3. Un analyseur profond et non déterministe basé sur les Grammaires de Propriétés

Contrairement aux deux autres algorithmes, nous observons ici une plus grande dispersion des résultats.

La complexité théorique de cet algorithme est exponentielle, mais son fonctionnement et sa manière d'être contraint en permanence par la grammaire lui confèrent un fonctionnement moyen réellement polynômial, même si l'analyse de deux phrases de taille identique pourra donner des temps d'analyse très différents. Ainsi, si Nm est le nombre de mots d'une phrase, la meilleure évaluation de sa complexité correspond à un polynôme d'ordre 2,4 (Nm2.4*Constant).

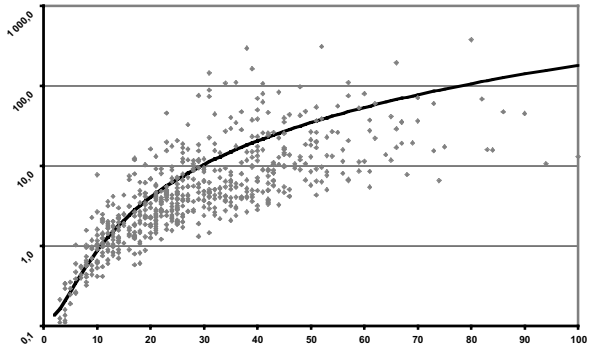


Figure 5
Million d'instructions / nombre de mots
pour l'analyseur profond (échelle logarithmique)

4. Sélection de granularité entre techniques. Deux expériences d'évaluation, un programme pour réaliser cette opération

4.1. Comment évaluer empiriquement les analyseurs sans corpus de référence

La question d'évaluer des analyseurs (même superficiels) est un problème en soi. À la différence de l'étiquetage lexical (POS-tagging), beaucoup d'aspects peuvent changer d'un système à l'autre, y compris les sorties de l'analyseur elles-mêmes.

D'une manière générale, évaluer un système consiste à comparer pour une entrée donnée sa sortie à un résultat de référence, normalisé. Dans le cas de l'analyse syntaxique, la

référence est un treebank, la comparaison se fait en comparant les parenthésages respectifs de l'analyseur et du Treebank. Ceci nécessite d'abord la disponibilité d'un treebank (une telle ressource existe seulement pour peu de langues) et signifie également que l'analyseur doit produire le même type d'information que celui du corpus de référence. Ce point peut être problématique. D'abord, parce que le parenthésage n'est pas totalement libre de toute théorie, ensuite, parce qu'une telle ressource indique habituellement une seule solution. Enfin, comme expliqué ci-dessus, le parenthésage n'est pas la seule information que nous voudrions évaluer.

De plus, il nous paraît intéressant de ne pas limiter une évaluation à la seule comparaison des différentes sorties : il est également nécessaire, afin d'interpréter une telle comparaison, de donner quelques indications sur les ressources et les techniques employées par le système. Par exemple, il est important d'avoir des indications sur :

- La couverture lexicale :
    - Nombre d'entrées
    - Représentation (propriétés lexicales)
- La couverture syntaxique :
    - Nombre de catégories
    - Différents phénomènes syntaxiques
- La stratégie d'analyse :
    - Robustesse
    - Efficacité
    - etc.

Nous proposons ici d'extraire des informations d'évaluation à partir d'une technique de comparaison. En d'autres termes, nous montrons que comparer différents analyseurs, à condition que la méthode soit systématique, permet dans certains cas de donner des éléments d'évaluation.

4.1.1. Évaluer et/ou multiplexer

L'idée de retrouver les mêmes frontières ou parenthésages par le biais de différents analyseurs syntaxiques nous a conduits à imaginer un programme susceptible de fusionner, à l'aide de paramètres, les sorties de différents analyseurs. Un des buts sous-tendus par cette technique est l'élimination des parenthésages erronnés et la conservation des meilleurs (voir figure ci-dessous).

Cette technique devant traiter des ensembles de frontières, les paramètres employés sont de deux sortes :

- Des opérateurs ensemblistes
  - -union,
  - -intersection,
  - -complément.
- Des poids donnant plus ou moins d'importance à tel analyseur plutôt qu'à tel autre pour une catégorie donnée. Afin d'éviter les erreurs connues d'un analyseur ou au contraire donner la priorité à un analyseur.

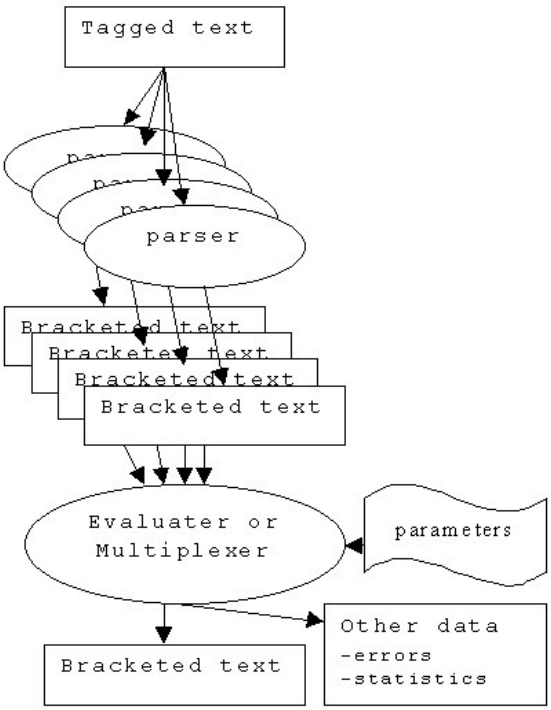


Figure 6
Un programme d'évaluation ou et de multiplexage

En l'absence de corpus de référence, nous ne pouvons que comparer les sorties de différents analyseurs et effectuer une expertise manuelle sur les résultats afin d'affiner les paramètres du multiplexeur. On peut aisément imaginer que ce programme fonctionnerait à merveille si nous disposions d'un tel corpus de référence. Nous pourrions aussitôt injecter ce corpus en entrée du multiplexeur avec un poids maximal pour toutes ses catégories et nous obtiendrions une évaluation des qualités et des défauts de nos analyseurs.
Mais tous les paramètres utilisables par ce programme ne peuvent pas être trouvés sans une bonne évaluation des sorties de chaque analyseur. Ces deux besoins (paramètres et évaluation) sont si proches qu'il est difficile de les distinguer.
C'est seulement par une méthode empirique d'évaluation et de paramétrage, puis de réévaluation et de reparamétrage (de manière rétroactive allant de l'évaluation au paramétrage puis des paramètres affinés à une nouvelle évaluation), que nous pouvons réaliser un programme d'évaluation valide pour un jeu d'analyseurs donnés.

4.1.2. Intérêt d'un tel programme

À la lecture de ce qui précède, il paraît fastidieux de travailler avec un tel programme. Retenons cependant que seul le paramétrage doit être réalisé manuellement. Le reste du travail, à savoir l'extraction des frontières communes, est automatique.
L'intérêt de ce programme apparaît alors si on se place du point de vue de la problématique initiale : nous ne voulons pas exclure des analyses apparemment contradictoires, mais au contraire les traiter comme des interprétations en concurrence pour un même texte. C'est exactement ce que fait ce programme. Dans cette optique, son avantage est de fournir une *sélection de granularité* parmi plusieurs techniques. Une réponse directe à notre problématique. Des expériences qui démontreront l'utilité et la pertinence de cette approche restent à effectuer.

4.2. Expériences

L'évaluation présentée ici repose sur deux expériences menées sur un corpus de 13236 phrases étiquetées extraites du journal *Le Monde,* nous les devons au projet CLIF (voir http://www.talana.linguist.jussieu.fr et [Abe03]).
Deux types d'étiquetage des catégories lexicales ont été employés : l'original (projet CLIF) et celui obtenu en utilisant l'étiqueteur automatique WinBrill.
Ces choix découlent d'une volonté de réaliser nos expériences d'une part sur des phrases courantes et d'autre part sur des textes étiquetés automatiquement.

L'avantage de disposer d'un corpus étiqueté (au niveau lexical) par des experts a fourni l'idée de la première expérience.

Afin de mieux appréhender ce qui peut être déterminé par notre multiplexeur, nous n'utiliserons comme paramètres que l'opérateur d'intersection et le même poids pour chaque sortie d'analyseur. D'autres travaux proposeront des analyses plus fines.

Dans les expériences qui suivent, les algorithmes proposés à l'étude sont :

A1 l'analyseur Chink/Chunk

A2 l'analyseur superficiel décrit plus haut

A3 un autre analyseur superficiel qui n'est pas détaillé ici.
Cet analyseur utilise aussi les Grammaires de Propriétés

A4 l'analyseur profond décrit plus haut.

4.2.1. Première expérience

La première expérience se propose de comparer les **frontières des chunks**, en fonction de **l'algorithme** et de **l'étiquetage**. Pour ce faire, nous avons simplifié massivement la sortie générée par nos programmes A2, A3 et A4 afin de n'en conserver que les chunks, sacrifiant par là les données linguistiques et les informations hiérarchiques pour ne disposer que d'analyses directement comparables avec celles de A1.

La question à laquelle cette expérience doit répondre est la suivante : l'étiquetage lexical perturbe-t-il l'analyse syntaxique ?

Le protocole expérimental est décrit par la figure ci-dessous :

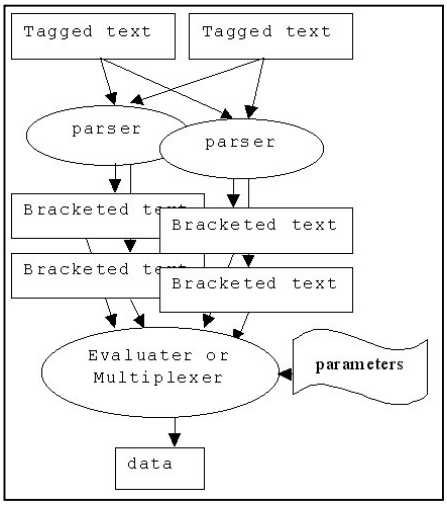


Figure 7
Protocole pour la première expérience

Chaque analyseur est déclenché sur chacun des corpus (jeu d'étiquettes CLIF et jeu d'étiquettes BRILL). Les données à évaluer constituent donc 8 fichiers correspondant aux 4 analyseurs. Déclencher le multiplexeur sur ces fichiers nous obligerait à confronter deux à deux ces fichiers, soit 28 confrontations à réaliser.
Comme cette expérience n'a pour but que d'évaluer l'influence de l'étiquetage sur l'analyse, nous n'exposerons ici que les résultats de la comparaison de A1 avec A2.

Algorithme	Mots par chunk
A1 Humain	3.49
A1 avec WinBRILL	3.34
A2 Humain	2.02
A2 avec WinBRILL	1.96

Figure 8
Résultats de la première expérience : mots par chunk

La table ci-dessous donne le nombre de frontières communes pour chaque multiplexage réalisé (après avoir supprimé toute information hiérarchique des fichiers testés).

1ère évaluation	A1 Humain	A1 WinBRILL	A2 Humain	A2 WinBrill
A1 Humain	100%	92%	82%	77%
A1 WinBRILL	89%	100%	75%	81%
A2 Humain	50%	47%	100%	82%
A2 WinBRILL	45%	50%	80%	100%

Figure 9
Résultats de la première expérience : frontières communes pour deux méthodes d'analyse

Cette table peut être lue de la façon suivante :
* Les frontières communes à A1 avec un étiquetage humain et A2 avec étiquetage humain, représentent 82% du nombre de frontières fournies par A1
* Les frontières communes à A2 avec un étiquetage humain et A1 avec étiquetage humain, représentent 50% du nombre de frontières fournies par A2

Les résultats sont instructifs, mais deux variables doivent être contrôlées :
* La différence de comportement du même algorithme avec deux jeux d'étiquetage,
* La différence entre deux algorithmes donnés, dont les sorties sont très différentes.

Il ressort de cette expérience que les différences de frontières d'un même analyseur pour deux étiquetages sont de 2 à 3 %, ce qui indique que l'étiquetage automatique reste pertinent pour la notion de frontière face à un étiquetage expert.
Ce résultat est éclairé par une autre information : le nombre de mots par chunk (table précédente).
Une seconde conclusion est que les algorithmes sont sensibles à la qualité de l'étiquetage (c.à.d. qu'ils réagissent à la variabilité) : ces résultats indiquent que A1 perd jusqu'à 10% de ses frontières pour un étiquetage non humain et A2 jusqu'à 20%.
Un dernier point est que A1 et A2 ont réellement de 47% à 82% de frontières communes. À la suite de ce qui a été énoncé plus haut, nous pouvons utiliser les frontières communes afin d'harmoniser et de garantir la qualité des différents sorties. C'est ce point que nous discuterons dans la seconde expérience.

4.2.2. Deuxième expérience

Cette expérience a pour objectif de comparer les trois approches basées sur les Grammaires de Propriétés. Les frontières communes seront comparées catégorie par catégorie.

Algorithm	A2	A3	A4
Chunks/ sentence	15.03	19.04	18.97
Words/chunk	1.90	1.50	1.50

Figure 10
Statistiques de la seconde expérience

NP	A2	A3	A4
A2	100%	54%	45%
A3		100%	100%
A4			100%

Figure 11
SN frontières communes

VP	A2	A3	A4
A2	100%	29%	27%
A3		100%	75%
A4			100%

Figure 12
SV frontières communes

AP	A2	A3	A4
A2	100%	50%	43%
A3		100%	86%
A4			100%

Figure 13
SA frontières communes

PP	A2	A3	A4
A2	100%	57%	49%
A3		100%	85%
A4			100%

Figure 14
SP frontières communes

COORD	A2	A3	A4
A2	-	0%	-
A3		100%	0%
A4			-

Figure 15
COORD frontières communes

Les approches A2 et A3 sont très différentes (48% de frontières communes en moyenne). Ceci est dû en partie aux différences entre les jeux d'étiquettes syntaxiques employés : A3 utilise des catégories que A2 ne connaît pas. Plus précisément, les SN, SA, SP et SV ont respectivement 55%, 50%, 57% et 30% de frontières communes.

A3 est plus proche de A4, qui cherche à satisfaire toutes les contraintes (90% de couverture).Les SN, SA, SP et SV ont respectivement 100%, 85%, 86%, 71% de frontières communes.

Ces résultats impliquent deux conclusions :

* Les frontières communes nous informent sur l'originalité ou le conformisme d'un analyseur par rapport à un autre.

* La simple connaissance de ce que fait un analyseur nous permet de paramétrer les opérateurs ensemblistes et leurs poids.

Un guide de lecture pour ces tables peut se trouver dans le fait que l'algorithme A4 a donné les meilleurs résultats en comparaison à une évaluation experte de 10 phrases. Il vient que la plupart des frontières communes que A4 partage avec A3 et A2 doivent porter un grand poids et doivent être multiplexées avec un opérateur d'intersection.

Une autre information réside dans le fait que A3 connaît des catégories qu'ignorent A2 et A4 (cf. table COORD). Cette connaissance implique que la catégorie COORD de A3 doit être incluse avec un poids de 100% et un opérateur d'union.

5. Conclusions et perspectives

Nous avons démontré à travers les expériences présentées dans cet article qu'il est possible de calculer efficacement différents niveaux de structure syntaxique pour une phrase, en utilisant les mêmes ressources linguistiques. Le formalisme basé sur la résolution de contraintes proposé ici rend possible la sélection de granularité, depuis une simple détection des frontières jusqu'à une analyse profonde et non déterministe, via une analyse superficielle et déterministe. Nous définissons ainsi la notion de *technique à granularité variable*.

Un autre résultat intéressant apparaît dans la perspective de combiner, de multiplexer, les résultats des différentes approches à l'aide d'opérateurs logiques. De fait, nous avons observé que cette technique, appliquée aux frontières communes obtenues par nos analyseurs, éliminait les groupes mal formés ainsi que les moins remarquables. Tout en accroissant la taille des blocs obtenus, les informations linguistiques disponibles sont préservées, ce qui reste un des principaux problèmes pour les systèmes de synthèse de parole. Enfin, cette technique offre une granularité paramétrable par la connaissance de l'importance relative des techniques en compétition. Nous définissons par cette technique la *sélection de granularité entre techniques*.

La technique de multiplexage reste à améliorer pour les opérateurs ensemblistes de disjonction et d'exclusion. D'autres techniques sont envisagées ainsi que la formalisation logico-mathématique des Grammaires de Propriétés dans [Van&Al03]

La possibilité de régler le niveau de granularité en fonction des données à traiter ou de l'application concernée par ce traitement se présente comme un réel progrès, mais il reste encore à déterminer dans quel contexte et de quelle manière le contexte peut dicter, diriger, cette sélection de granularité. Des éléments de réponse sont déjà sensibles dans les travaux réalisés et une formalisation systématique de ce problème est en développement.

Références

[Abe03] ABEILLE, A. (2003), *Une grammaire électronique du français*, CNRS Editions, Paris.

[Abn91] ABNEY, S. (1991), Parsing by chunks, *in* Berwick, R., Abney, S., Tenny, C. (eds). *Principle-based Parsing,* Kluwer Academic Publishers, p. 257-278.

[Abn96] ABNEY, S. (1996), Partial Parsing via Finite-State Calculus, *Proceedings of ESSLLI'96,* Robust Parsing Workshop.

[Abn97] ABNEY, S. (1997), Part-of-speech tagging and partial parsing, *in* Young, S., Bloothooft, G. *Corpus-Based Methods in Language and Speech Processing,* Kluwer Academic Publishers, Dordrecht, p. 118-136.

[All&Al79] ALLEN, J.; HUNNINCUTT, S.; CARLSON, R.; GRANSTRÖM, B. (1979), MITalk-79 : The 1979 MIT text-to-speech system, *in* Wolf and Klatt (eds), *Speech Communications, 97th Meeting of the ASA*, p. 507-510.

[All&Al87] ALLEN, J.; HUNNINCUTT, S.; KLATT, D. (1987), *From text to speech : The MITalk system,* Cambridge University Press.

[ArcLan87] ARCHANGELI, D.; LANGENDOEN, D.T. (eds) (1997), *Optimality Theory,* Blackwell.

[BèsBla99] BES, G.; BLACHE, P. (1999), Propriétés et analyse d'un langage, *Proceedings of TALN'99.*

[BlaMor90] BLACHE, P.; MORIN, J.-Y. (1990), Bottom-up Filtering : a Parsing Strategy for GPSG, *Proceedings of COLING'90.*

[Bla00] BLACHE, P. (2000), Constraints, Linguistic Theories and Natural Language Processing, *in* Christodoulakis, D. (ed.), *Natural Language Processing,* LNAI 1835, Springer-Verlag.

[BlaBal01] BLACHE, P.; BALFOURIER, J.-M. (2001), Property Grammars : a Flexible Constraint-Based Approach to Parsing, *Proceedings of IWPT-2001.*

[BlaVan02] BLACHE, P.; VAN RULLEN, T. (2002), An evaluation of different symbolic shallow parsing techniques, *Proceedings of LREC-02.*

[Bla&Al02] BALFOURIER, J.-M.; BLACHE, P.; VAN RULLEN, T. (2002), From Shallow to Deep Parsing Using Constraint Satisfaction, *Proceedings of COLING'2002,* p. 36-42.

[Bou&Al01] BOUMA, G.; MALOUF, R., SAG, I. (2001), Satisfying Constraints on Extraction and Adjunction, *in* Maling, J., *Natural Language and Linguistic Theory,* 19:1, Kluwer, Dordrecht, p. 1-65.

[Cha00] CHANOD, J.-P. (2000), Robust Parsing and Beyond, *in* Van Noord, G.; Junqua, J.-C. (eds), *Robustness in Language Technology,* Kluwer, Dordrecht.

[ChaTap96] CHANOD, J.-P.; TAPANAINEN, P. (1996), A non deterministic Tokenizer for Finite State Parsing, *Proceedings of the ECAI 96 Workshop.*

[ChaMok96] CHANOD, J.-P., MOKHTAR. (1996), *Incremental Finite State Parsing,* Report, Rank Xerox research center.

[ChaTap97] CHANOD, J.-P.; TAPANAINEN, P. (1997), *A Robust Finite State Parser,* Report, Rank Xerox research center.

[DiC98] DI CRISTO, P. (1998), *Génération automatique de la prosodie pour la synthèse à partir du texte*, Thèse de doctorat, Université de Provence, France.

[DiC&Al00] DI CRISTO, A.; DI CRISTO, P.; CAMPIONE, E; VERONIS, J. (2000), A prosodic model for text to speech synthesis in French, *in* Botinis, A. (ed.), *Intonation, Analysis, Modelling and Technology,* Kluwer, Dordrecht.

[DucDeb01] DUCHIER, D.; DEBUSMANN, R. (2001), Topological Dependency Trees : A Constraint-Based Account of Linear Precedence, *Proceedings of ACL.*

[Gri&Al95] GRINBERG, D.; LAFFERTY, J., SLEATOR, D. (1995), *A robust parsing algorithm for link grammars,* CMU-CS-95-125, Carnegie Mellon University.

[Gre96] GREFENSTETTE, G. (1996), *Light parsing as finite State Filtering*, Rank Xerox research center. Workshop on Finite State Models of Language ECAI96 Budapest.

[Kar90] KARLSSON, F. (1990), Constraint Grammar as a framework for Parsing Running Text, *Proceedings of COLING'90.*

[Kin02] KINYON, A.; PROLO, C. (2002), A Classification of Grammar Development Strategies, *Proceedings of Workshop on Grammar Engineering and Evaluation (COLING-02).*

[KübHin01] KÜBLER, S.; HINRICHS, E. (2001), From Chunks to Function-Argument Structure : A similarity-Based Approach, *Proceedings of ACL-01.*

[LibChu92] LIBERMAN, M.; CHURCH, K. (1992), Text analysis and word pronunciation in text-to- speech synthesis, *in* Furui, S., Sondhi, M.M. (eds), *Advances in Speech Signal Processing,* Dekker, p. 791-831.

[Mar90]MARUYAMA, H. (1990), Structural Disambiguation with Constraint Propagation', *Proceedings of ACL'90.*

[Pla&Al00] PLA, F.; MOLINA, A.; PIETRO, N. (2000), *Tagging and Chunking with Bigrams,* Universitat Politecnica de València.

[PolSag94] POLLARD, C.; SAG, I. (1994), *Head-driven Phrase Structure Grammars,* CSLI, Chicago University Press.

[PriSmo93] PRINCE, A.; SMOLENSKY, P. (1993), *Optimality Theory : Constraint Interaction in Generative Grammars,* Technical Report RUCCS TR2, Rutgers Center for Cognitive Sciences.

[SagWas99] SAG, I.; WASOW, T. (1999), *Syntactic Theory. A Formal Introduction,* CSLI, Chicago University Press.

[Van&Al03] VAN RULLEN, T.; GUENOT, M.-L.; BELLENGIER, E. (2003), Formal representation of Property Grammars, *Proceedings of ESSLLI 2003.*